



Modificando o app para um projeto postos de gasolina

App postos de gasolina

Primeiramente vamos criar nosso formulário de cadastro de usuários vamos criar um componente e começar pelo HTML

```
<ion-content  color="primary">
  <form  [formGroup]="userForm" (ngSubmit)="onSubmit()" >
    <ion-grid>
      <ion-row color="primary" justify-content-center>
        <ion-col class="ion-text-center" size-md="6" size-lg="5" size-xs="12">
          <div text-center>
            <h3>Cadastre-se</h3>
          </div>
          <div class="ion-padding">
            <ion-item>
              <ion-input  name="name" type="text" placeholder="Nome" formControlName="name"  required></ion-input>
            </ion-item>
            <ion-item>
              <ion-input name="email" type="email" placeholder="your@email.com" formControlName="email"  required>
```



```
</ion-input>

          </ion-item>

          <ion-item>

            <ion-input name="phone" type="text" placeholder="phone"
formControlName="phone" required></ion-input>

          </ion-item>

          <ion-item>

            <ion-input name="password" type="password"
placeholder="Senha" formControlName="password"  required></ion-
input>

          </ion-item>

        </div>

        <div class="ion-padding">

          <ion-button  size="large" type="submit"
[disabled]="!userForm.valid" expand="block">Cadastrar</ion-
button>

        </div>

      </ion-col>

    </ion-row>

  </ion-grid>

 </form>

</ion-content>
```

Modificamos a tag form para ela receber o formGroup.

```
<form  [formGroup]="userForm" (ngSubmit)="onSubmit()" >
```

Agora vamos no nosso componente e adicionar o formGroup.



```
user: User = new User('','','','','');

userForm = this.fb.group({
  name: ['', Validators.required],
  email: ['', [Validators.required, Validators.email]],
  phone: ['', Validators.required],
  password: ['', Validators.required],
});
```

Vamos criar também uma classe User.

```
export class User {
  constructor(
    public id: string,
    public name: string,
    public phone: string,
    public email: string,
    public password: string) {
  }
}
```

Agora criamos nosso construtor

```
constructor(
  private fb: FormBuilder,
  private db: DbService,
```

```
  private service: UserService) {}
```



No module do componente vamos fazer as seguintes alterações.

```
@NgModule({
 imports: [
  CommonModule,
  FormsModule,
  IonicModule,
  HttpClientModule,
  ReactiveFormsModule,
  RegisterPageRoutingModule
 ],
 declarations: [RegisterPage],
 providers: [UserService, DbService]
})
export class RegisterPageModule {}
```

Adicionamos as bibliotecas HttpClienteModule e ReactiveFormsModule, pois o formulário que estamos criando será reativo.

Agora criamos a função para salvar os dados.

```
 onSubmit(): void {
  if(this.userForm.controls['name'].value)
   this.user.name= this.userForm.controls['name'].value;
```

```
if(this.userForm.controls['email'].value)
  this.user.email= this.userForm.controls['email'].value;
if(this.userForm.controls['phone'].value)
  this.user.phone = this.userForm.controls['phone'].value;
if(this.userForm.controls['password'].value)
  this.user.password = this.userForm.controls['password'].value;
this.db.addUser(this.user);
```



Primeiramente vamos gravar os dados no SQLite e posteriormente vamos gravar na Rest API.

Para criar nosso banco de dados vamos alterar nosso componente principal para executar o SQL e criar o banco caso ele ainda não esteja criado.

```
const SQL = 'CREATE TABLE IF NOT EXISTS user(id INTEGER PRIMARY KEY, name TEXT NOT NULL,email TEXT NOT NULL,'+
  'phone TEXT NOT NULL,password TEXT NOT NULL);'+
  'CREATE TABLE IF NOT EXISTS gas_station ( id INTEGER PRIMARY KEY, title TEXT NOT NULL,'+
  'address TEXT NOT NULL, address TEXT NOT NULL, number TEXT NOT NULL, city TEXT NOT NULL,lat TEXT NOT NULL, lng TEXT NOT NULL);'+
  'CREATE TABLE IF NOT EXISTS gas (id INTEGER PRIMARY KEY AUTOINCREMENT, posto_id INTEGER,'+
  'Name TEXT NOT NULL,Price NUMERIC,FOREIGN KEY(gas_station_id) REFERENCES gas_station(id));';

constructor(private sqlite: SQLite) {}

  ngOnInit() {
```



```
  this.sqLiteGenerateDB();
}
sqLiteGenerateDB(){

  this.sqlite.create({
    name: 'data.db',
    location: 'default'
  })
    .then((db: SQLiteObject) => {
      console.log(SQL);
      db.executeSql(SQL, [])
        .then(() => console.log('Executed SQL'))
        .catch(e => console.log(e));
    })
    .catch(e => console.log(e));
}
```

A função acima executa o SQL e caso o banco não tenha sido criado ele cria o mesmo e posteriormente as tabelas.

Arquivo de Serviços do Banco de Dados SQLite

Vamos criar um service para manipular o banco de dados e agora vamos começar a configuração do mesmo.

```
export class DbService {
```



```
private storage: any;

userList = new BehaviorSubject([]);

private isDbReady: BehaviorSubject<boolean> = new
BehaviorSubject(false);

constructor(

  private platform: Platform,

  private sqlite: SQLite,

  private httpClient: HttpClient,

) {

  this.platform.ready().then(() => {

    this.sqlite.create({

      name: 'data.db',

      location: 'default'

    })

    .then((db: SQLiteObject) => {

        this.storage = db;

    });

  });

}

dbState() {

  return this.isDbReady.asObservable();

}
```

Agora vamos adicionar a função adicionar usuário e posto de gasolina.

```
addUser(user: User) {
  let data = [user.name, user.email, user.phone, user.password];
  return this.storage.executeSql('INSERT INTO user (name, email,
phone, password) VALUES (?, ?, ?, ?)', data)
  .then((res: any) => {
    console.log(res)
    this.getUsers();
  });
}

addGas(gas: GasStation) {
  let data = [gas.title, gas.address, gas.number, gas.city, gas.lat,
gas.lng];
  return this.storage.executeSql('INSERT INTO gas_station (title,
address, number, city, lat, lng) VALUES (?, ?, ?, ?, ?, ?)', data)
  .then((res: any) => {
    console.log(res)
    this.getGas();
  });
}
```

App postos de gasolina página de Login

Por ser um projeto somente de teste, vamos fazer o login diretamente no SQLite, para isso vamos criar nosso HTML, porém

esse formulário não vamos usar o formGroup. Vamos usar o ngModel para mais informações sobre formulários. Você poc ncontrar no site oficial do Angular.

Para dar início criamos um componente denominado login e vamos começar pelo HTML:

```
<ion-content color="primary" padding>

 <form #form="ngForm" (ngSubmit)="onSubmit(form)" >

  <ion-grid>

   <ion-row color="primary">

    <ion-col class="ion-text-center" size-md="6" size-lg="5" size-
xs="12">

     <div text-center>

      <h3>Login</h3>

     </div>

     <div class="ion-padding">

      <ion-item>

       <ion-input name="email" type="email"
placeholder="your@email.com" ngModel required></ion-input>

      </ion-item>

      <ion-item>

       <ion-input name="password" type="password"
placeholder="Password" ngModel required></ion-input>

      </ion-item>

     </div>
```



```
        <div class="ion-padding">

          <ion-button size="large" type="submit"
[disabled]="form.invalid" expand="block">Login</ion-button>

        </div>

      </ion-col>

    </ion-row>

    <ion-row>

      <div text-center>

    Caso não tenha conta,  <a routerLink='/register'>

      cadastre-se</a>!

      </div>

    </ion-row>

    </ion-grid>

  </form>

</ion-content>
```

Agora vamos editar o componente.

```
constructor(

  private router: Router,

  private db: DbService) { }

 onSubmit(form:any){

  console.log(form.controls.email.value)

  this.db.login(form.controls.email.value).then(res => {
```



```
this.router.navigate(['/app/tabs/tab1'])
```

A função onSubmit vai executar a função login no banco de dados que vai verificar se aquele email está cadastrado no banco de dados.

Agora vamos alterar nosso DBservice para criar a função de login

```
mail: string): Promise<User

return this.storage.executeSql('SELECT * FROM user WHERE email = ?', [email]).then((res: any) =>

return
```

Agora podemos deixar como exercício a criação do cadastro de combustíveis, para isso vamos criar um componente denominado fuel, podemos reaproveitar os códigos anteriores para criar esse componente. No material de última aula podemos encontrar o github

do projeto completo e podemos verificar como fica o cadastro de combustíveis.

App postos de gasolina adicionando GPS.

Agora vamos usar os conhecimentos dos capítulos anteriores para pegar a localização de GPS durante o cadastro do posto de gasolina.

Primeiro vamos adicionar a função para pegar a localização.

```
  return Geolocation
  =>
   return coordinates

  =>
   throw(e
```

Agora vamos criar essa função em nosso ngOnInit.

```
async
 try
  const coordinates = await this
  console.log('Current position:', coordinates
  if (coordinates?.coords

this.gasForm.controls['lat'].setValue(coordinates
```



Ele vai preencher automaticamente a latitude e longitude em nosso formulário, vamos continuar modificando o App na próxima aula.

Atividade Extra

Para aprofundar o conteúdo assista o vídeo a seguir que é um exemplo de uma App Ionic com mapas.

Canal: Ronan Adriel Zenatti

Título a ser buscado no youtube: **Ionic com Google Maps - 02 - Exibindo Mapa no aplicativo**

Referência Bibliográfica

ANGULAR DOCS. **Angular IO**. Disponível em: <https://angular.io>. Acesso em: 30 nov. 2022.

IONIC FRAMEWORK. **IONIC**. Disponível em: <https://ionicframework.com>. Acesso em: 30 nov. 2022.

CAPACITOR FRAMEWORK. **CAPACITOR.** Disponível em: <https://capacitorjs.com/docs>. Acesso em: 30 nov. 2[illegible].

Atividade Prática 15 – Cadastros

Título da Prática: Criar uma CRUD de professores ou de faculdades

Objetivos: Utilizar SQLite ou Rest API para criar um CRUD diferente do que vimos durante as aulas.

Materiais, Métodos e Ferramentas: Iremos fazer uma pesquisa na internet e utilizar os conceitos de cadastros utilizando IONIC

Atividade Prática

Utilizando os conceitos das aulas criar um CRUD usando SQLite ou Rest API usando Ionic criar um objeto diferente do que estamos estudando na disciplina.

Gabarito Atividade Prática

Primeiro vamos fazer o formulário

```
<ion-header>
  <ion-toolbar>
    <ion-title>professor</ion-title>
  </ion-toolbar>
</ion-header>
```



```
<ion-content color="primary" padding>

 <form #form="ngForm" (ngSubmit)="onSubmit(form)" >

  <ion-grid>

   <ion-row color="primary">

    <ion-col class="ion-text-center" size-md="6" size-lg="5" size-
xs="12">

     <div text-center>

      <h3>Registro de professor</h3>

     </div>

     <div class="ion-padding">

      <ion-item>

       <ion-input name="name" type="text" placeholder="Digite o
nome" ngModel required></ion-input>

      </ion-item>

      <ion-item>

       <ion-
input name="email" type="email" placeholder="your@email.com" ngModel required>
</ion-input>

      </ion-item>

      <ion-item>

       <ion-input name="materia" type="text" placeholder="Digite a
materia do professor" ngModel required></ion-input>

      </ion-item>
```



```
        </div>

        <div class="ion-padding">

          <ion-
button size="large" type="submit" [disabled]="form.invalid" expand="block">Cadastr
button>

        </div>

      </ion-col>

    </ion-row>

  </ion-grid>

</form>

</ion-content>
```

Agora vamos criar o componente de cadastro

```
import { Component, OnInit } from '@angular/core';

import { Router } from '@angular/router';

import { DbService } from '../services/db.service';

@Component({

 selector: 'app-professor',

 templateUrl: './professor.page.html',

 styleUrls: ['./professor.page.scss'],

})

export class ProfessorPage implements OnInit {

 constructor(

  private router: Router,
```

```
  private db: DbService) { }

 ngOnInit() {

 }

 onSubmit(form:any){

  console.log(form.controls.email.value)

  this.db.addUser(form).then((res:any) => {

   console.log(res)

   console.log('-----------------<entrou>----------1')

  })

 }

}
```



Agora adicionamos uma função ao arquivo de banco de dados

```
 addTeacher(user: User) {

  let data = [user.name, user.email, user.phone, user.password];

  return this.storage.executeSql('INSERT INTO teacher (name, email,
materia) VALUES (?, ?, ?)', data)

   .then((res: any) => {

    console.log(res)

    return res;

   });

 }
```

Ir para exercício